POSTS AND TELECOMMUNICATIONS INSTITUTE OF TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY I

o0o

ASSIGNMENT 1 REPORT

PYTHON PROGRAMMING LANGUAGE

Instructor:	**Kim Ngoc Bach**
Student:	**Phung Thu HuoHuong**
Student ID:	**B23DCVT201**
Class:	**D23CQCEO6-B**
Academic Year:	**2023 - 2028**
Training System:	**Full-time University**

Hanoi, 2025

POSTS AND TELECOMMUNICATIONS INSTITUTE OF TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY I

o0o

ASSIGNMENT 1 REPORT

PYTHON PROGRAMMING LANGUAGE

Instructor:	**Kim Ngoc Bach**
Student:	**Phung Thu HuongHuong**
Student ID:	**B23DCVT201**
Class:	**D23CQCEO6-B**
Academic Year:	**2023 - 2028**
Training System:	**Full-time University**

Hanoi, 2025

INSTRUCTOR'S COMMENTS

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Score: **(In words:**)

Hanoi, day month year 20...

Instructor

Contents

List of Figures

Introduction

This report is conducted to fulfill the requirements of Major Assignment 1 within the Python Programming course. The focus of the assignment is to apply data collection, analysis, and modeling techniques to shed light on the performance of players in the English Premier League 2024-2025 season.

The primary data source used in this study is the fbref.com website, which provides detailed statistics on player performance. The data collection scope includes all players who have played more than 90 minutes in the English Premier League 2024-2025 season. Additionally, estimated transfer value data for players who have played over 900 minutes was collected from the footballtransfers.com website. The entire data collection process was carried out on May 2, 2025. Therefore, all analyses and results presented in this report reflect the situation and data of the players up to this specific point in the season.

The content of the report is organized into four main chapters, as follows:

Chapter 1. Player Data Collection from fbref.com: This chapter details the process of collecting player statistics from the fbref.com website using automation tools such as Selenium and BeautifulSoup. The initial data cleaning and storage steps are also described in detail. This chapter addresses the requirements of Part I of the assignment.

Chapter 2. Data Analysis and Visualization: This chapter focuses on the descriptive analysis of the collected data. The analyses include identifying the top 3 highest-performing and bottom 3 lowest-performing players for each statistic, calculating summary statistics (mean, median, standard deviation) for each statistic across the entire league and for each team. Furthermore, this chapter presents histogram plots illustrating the data distribution and identifies the best-performing team based on statistical indicators.

Chapter 3. Player Clustering using K-Means and PCA: This chapter applies the K-Means clustering algorithm to group players into clusters based on similar statistical characteristics. The PCA (Principal Component Analysis) dimensionality reduction technique is used to visualize the clusters on a two-dimensional plot, thereby clarifying the differences between player groups.

Chapter 4. Player Value Estimation: This chapter focuses on collecting estimated transfer value data for players from the footballtransfers.com website and proposes a method for estimating their market value. This method is based on building a machine learning model (Gradient Boosting Regressor) with features selected from performance statistics (collected in Chapter 1), basic player information, and historical transfer values. The feature selection, model training, and evaluation process are presented in detail.

Chapter 1

Collecting Player Data from fbref.com

In this section, I present a Python program to automatically collect statistical data of players competing in the English Premier League (EPL) 2024-2025 season from the website `https://fbref.com`. The indicators to be collected cover various aspects: nationality, age, playing time, match performance, passing, shooting, goalkeeping, etc.

The program is organized into Python module files:

- `football_stats_scraper.py`: main source code containing the entire process of automatically downloading, analyzing, and saving data.
- `config.py`: stores configurations such as the base URL, table identifier (HTML ID), list of indicators to collect (fields), columns to appear in the results (CSV_COLUMNS), path to the result file (CSV_FILE), etc.

After running, the program will generate a result file named:

- `results.csv`: contains players with total playing minutes > 90, already filtered and standardized, ready for further analysis.

The result file is saved at: `Report/OUTPUT_BAI1/results.csv`

1.1 Reasons for Choosing the Selenium Library

The fbref.com website is a modern website where statistical tables are not pre-loaded as static HTML but are generated via JavaScript after the page has been displayed. This prevents traditional libraries like `requests` or `urllib` from obtaining complete data because they only read the initial HTML code.

In this case, Selenium is the most suitable choice for the following reasons:

- **Complete JavaScript Rendering:** Selenium controls a real browser (like Chrome), so it can execute the entire page loading process, including JavaScript code that generates statistical tables.
- **Dynamic Interaction with DOM:** Selenium can wait (`wait`) for specific elements to appear before retrieving data. This is essential because fbref's statistical tables may take a few seconds to display after the page loads.

- **Scalability Support:** If future needs include clicking buttons, selecting different seasons, or navigating through pages, Selenium can handle this easily by simulating user behavior.

In summary, Selenium is an appropriate and powerful tool for collecting data from dynamic websites like fbref.com. When combined with BeautifulSoup, it provides both automation capabilities and easy HTML parsing.

1.2 Libraries Used

- `selenium`: controls the Chrome browser to automatically access and load data tables that use JavaScript.
- `BeautifulSoup4` (`bs4`) : parses the HTML DOM after the page has been fully rendered, helping to extract data based on `table_id` and `data-stat`.
- `pandas`: processes tabular data, converts from `dict` to `DataFrame`, handles missing values, and saves .csv files.
- `time`: uses `time.sleep()` to add delays between table loads, avoiding IP blocking.
- `os`: creates the output directory if it does not exist.
- `re`: used when special processing is needed (extracting IDs from URLs if further expanded).

1.3 Data Scraping Process (detailed in the source code)

1.3.1 Step 1: Read Table Configuration for Data Extraction

From the `config.py` file, the program retrieves the `TABLES` list, where each table is described as follows:

- `url`: specific path for each type of statistic (e.g., `/en/comps/9/stats/players/` for basic statistics).
- `table_id`: actual HTML ID of the table in the DOM (e.g., `stats_standard`, `stats_passing`...).
- `fields`: a list of indicators to collect, where each indicator is a pair: (column name in csv, `data-stat` name on fbref).

1.3.2 Step 2: Load Page Using Selenium

For each table, the program uses Selenium to:

- Access the corresponding URL (concatenated from `BASE_URL` + `table['url']` + `SEASON_SUFFIX`).
- Use `WebDriverWait` to wait until the table appears in the DOM.
- Use `driver.page_source` to retrieve the complete HTML code after the page has rendered.

1.3.3 Step 3: Analyze HTML using BeautifulSoup

- Use `BeautifulSoup(..., 'html.parser')` to parse the HTML.
- Find the `lt;table>` with `id = table`$_i d, then access$`lt;tbody>`$, and iterate through each$`lt;tr>$

- - `data-stat="player"` rightarrow player name
  - `data-stat="team"` rightarrow team name
  - and other statistics from `fields` rightarrow specific data such as goals, assists, playing time, etc.

Each player's data is stored with the key `(player`$_n ame, team_n ame) in a$`dictionary`.

1.3.4 Step 4: Combine data from multiple tables

After extracting all the tables, the program calls `combine`$_d ata(list_o f_d icts) to$:

- `Combine all data for each player (a player may appear multiple times if they played for multiple teams).`
- `If information for a player already exists, new statistics from other tables will be updated.`

1.3.5 Step 5: Filter qualified players (90 minutes)

- `Iterate through each row of the combined data.`
- `Take the "Playing Time:  minutes" column, remove commas, and convert it to an integer.`

- `Only keep players with a total number of minutes MIN`$_M INUTES(declared as 90 in$`config.p`

1.3.6 Step 6: Sort by first name

- `Use the get`$_f irst_n ame() function to get the player's first name (the part at the beginning of the full name$ *sorted alphabetically by first name.*

1.3.7 Step 7: Standardize data columns

- `Iterate through each row in the filtered list.`

- `For each column in CSV`$_C OLUMNS, if it's missing, fill in the default value "N/a"$.

1.3.8 Step 8: Export data to CSV file

- `Use pandas.DataFrame to convert the list of dict to a table.`

- `Use os.makedirs() to ensure the Report/OUTPUT`$_B AI1 directory exists. Write the file using$

1.4 Conclusion

- The total number of players collected and filtered in results.csv depends on the total playing time throughout the season, keeping only players 90 minutes.
- Statistical indicators include both basic and advanced metrics: Age, Nationality, Playing Time, Goals, Assists, xG, xAG, shooting parameters, passing parameters, goalkeeper parameters...
- The resulting file has a clear structure, suitable for the requirements of the problem, and serves as input for exercises 2 (statistics), 3 (KMeans clustering), and 4 (transfer value analysis).
- The program is reasonably separated, easy to read, and easy to expand. Simply updating the configuration in config.py allows for expansion to other seasons or leagues.

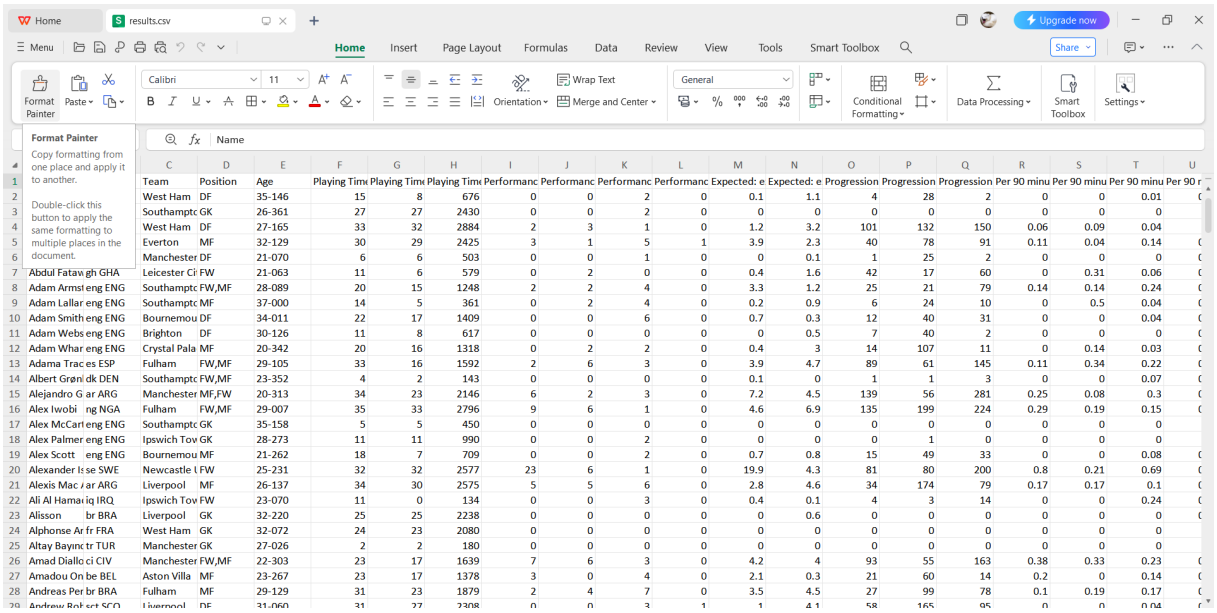


Figure 1.1: Illustration of results.csv file

Chapter 2

Data Analysis and Visualization

2.1 Analysis of Top 3 Highest and Lowest Players by Each Statistical Metric

2.1.1 Overview of the Problem Requirement

The problem requires processing a data file (results1.csv) containing statistical information of football players. The main objective is to identify and list the 3 players with the highest scores (Top 3) and the 3 players with the lowest scores (Bottom 3) for each statistical metric present in the data file. The results of this analysis are saved to a text file named top_3.txt.

2.1.2 Overview of the Main Code Logic (top_3.py)

The script top_3.py uses the pandas library to read data from the CSV file. Then, the script iterates through each statistical column (excluding basic information columns such as 'Name', 'Nation', 'Team', 'Position'). For each statistical column, the code attempts to convert the data to numeric format, removing invalid values (NaN). If the column can be considered numeric and has valid data, the code sorts the data to find the 3 highest and 3 lowest values along with the corresponding player names and teams. Finally, the results are formatted and written to the top_3.txt file.

2.1.3 Handling 'N/a' and Invalid Values

During data processing, 'N/a' values and their variations are considered missing data and are converted to NaN through the na_values parameter when reading the CSV file using Pandas. In addition, values that cannot be converted to numeric format are also coerced to NaN using pd.to_numeric(errors='coerce'). Instead of replacing missing values with a specific number (like 0), all rows containing NaN in the column being evaluated will be removed before ranking. This is done based on the following principles:

- Data Accuracy: NaN indicates that data is missing or not applicable, for example, a parameter specific to one position is not suitable for a player in another position. Replacing it with 0 would distort this nature.

- Avoiding Misinterpretation: Assigning a value of 0 to missing data can lead to misinterpretation - for example, mistakenly believing that a player did not score, while in reality the data was not recorded.
- Ensuring Fairness in Ranking: Considering only players with valid data helps the Top/Bottom 3 results accurately reflect actual performance, avoiding the inclusion of individuals with missing data in comparisons incorrectly.

Thus, keeping NaN values and removing missing values from the ranking process is a logical approach, ensuring objectivity and accuracy in the player data analysis.

2.1.4 Execution Steps

Initialization and Data Preparation:

- The script imports necessary libraries (pandas, os, numpy).
- Reads data from the results.csv file into a Pandas DataFrame, configured to handle variations of missing values as NaN.
- Identifies the list of statistical columns (stats_columns) to be analyzed by excluding identifier columns ('Name', 'Team', 'Position', 'Nation').

Processing Each Column and Writing Output File:

- Opens the top_3.txt file for writing results.
- The script iterates through each column col in stats_columns:
  - Write Column Title: Writes the current column name to the file.
  - Normalize & Filter Numeric Data: Uses pd.to_numeric(errors='coerce') to coerce the col column to numeric type, converting errors to NaN.
  - Creates a temporary DataFrame df_for_sort and removes rows with NaN values (dropna()) in this column. This step ensures that only valid numeric data is used for ranking.
  - Rank and Write Results:
    - If df_for_sort contains valid numeric data: Uses sort_values() to sort this DataFrame by the col column in descending order (finding Top 3) and ascending order (finding Bottom 3), then uses head(3).
    - The results (Name, Team, Metric) and data type information are written to the file.
  - Write Separator: Adds a line of ===... to separate the results between columns.

Completion: After processing all columns, the top_3.txt file is saved, containing all the analysis results.

```
--- Age ---
---------

Top 3 Player:
            Name     Team  Age
Łukasz Fabiański West Ham 40.0
    Ashley Young  Everton 39.0
    James Milner Brighton 39.0

Bottom 3 Player:
               Name           Team  Age
        Mikey Moore      Tottenham 17.0
Chidozie Obi-Martin Manchester Utd 17.0
 Myles Lewis-Skelly        Arsenal 18.0

================================================

--- Playing Time: matches played ---
----------------------------------

Top 3 Player:
           Name        Team  Playing Time: matches played
Youri Tielemans Aston Villa                             35
   Raúl Jiménez      Fulham                             35
     Bernd Leno      Fulham                             35

Bottom 3 Player:
         Name           Team  Playing Time: matches played
 Ayden Heaven Manchester Utd                              2
Billy Gilmour       Brighton                              2
         Neto    Bournemouth                              2

================================================
```

Figure 2.1: Image of the top_3.txt file results

2.1.5 Results (top_3.txt)

The top_3.txt file contains the analysis results for each statistical metric. Each metric is clearly presented with a title, followed by the Top 3 and Bottom 3 player lists.

2.2 Descriptive Statistics Calculation (median, mean, std dev) for Player Data

2.2.1 Introduction

This exercise requires writing a Python program to perform descriptive statistical calculations on player data. Specifically, the program needs to calculate:

- The median value for each statistical metric across the entire dataset.

- The mean and standard deviation for each statistical metric, calculated across the entire dataset and separately for each team.

The results are saved to the file results2.csv with a specific format, where the rows represent the entire dataset ('all') or a specific team, and the columns represent the statistical calculations (Median, Mean, Std) applied to each metric.

2.2.2 Library Selection

- pandas library: The primary library, indispensable for this task.
- os library: Used to manage file paths flexibly and independently of the operating system. os.path.join and os.path.dirname help accurately determine the location of the input file (results1.csv) and the output file (results2.csv without hardcoding paths.

2.2.3 Logic and Implementation Process (calculating_statistics.py)

The calculating_statistics.py program performs the following steps:

Step 1: Data Preparation and Reading:

- Use os to construct the paths to the input file results1.csv and the output file results2.csv.
- Read results.csv into a pandas DataFrame (df).
- A list na_values_list containing various representations of missing values ('N/a', 'n/a', '', ...) is provided to the na_values parameter of pd.read_csv to ensure correct identification of these values and their conversion to pandas NaN, facilitating subsequent calculations.

Step 2: Identifying Statistical Columns:

- Define a list exclude_cols containing columns that are not statistical data to be analyzed (e.g., 'Name', 'Nation', 'Team', 'Position', 'Age').
- Create a list stats_columns by filtering the columns in the DataFrame, keeping only those not present in exclude_cols.

Step 3: Calculating Overall Statistics ('all'):

- Select the columns in stats_columns from the DataFrame df.
- Use the .agg(['median', 'mean', 'std']) method to simultaneously calculate the median, mean, and standard deviation for each column in stats_columns.
- The returned result has statistical metrics as rows and original columns as columns. Use .T (transpose) to transpose it, making the original metric names (stats_columns) the row indices and 'median', 'mean', 'std' the column names, storing it in overall_stats. This facilitates easier access in the next step.

Step 4: Calculating Statistics by Group ('Team'):

- Use df.groupby('Team') to group the DataFrame by the values in the 'Team' column.
- On this GroupBy object, select the stats_columns.
- Apply .agg(['median', 'mean', 'std']) to calculate the statistical metrics for each team. The result (team_stats) will have a MultiIndex for both rows (Team) and columns (Statistic, Metric).

Step 5: Restructuring Data for Output:
Creating the 'all' row:

- Initialize an empty DataFrame all_row with the index 'all'.
- Iterate through each statistical column (col in stats_columns), creating new columns in all_row with names in the format f'Median of {col}', f'Mean of {col}', f'Std of {col}' and assigning the corresponding values taken from overall_stats calculated in Step 3.

Creating 'Team' rows:

- Initialize a DataFrame team_results with the team names as indices (taken from team_stats.index).
- Iterate through each statistical column (col in stats_columns), creating new columns in team_results with similar formatting as above. The values are taken from team_stats by accessing through the MultiIndex, for example: team_stats[(col, 'median')] to get the median column for the col metric.

Combining Results:

- Use pd.concat([all_row, team_results]) to concatenate the DataFrame containing the 'all' row and the DataFrame containing the rows for each team vertically, creating the final DataFrame final_results with the required structure.

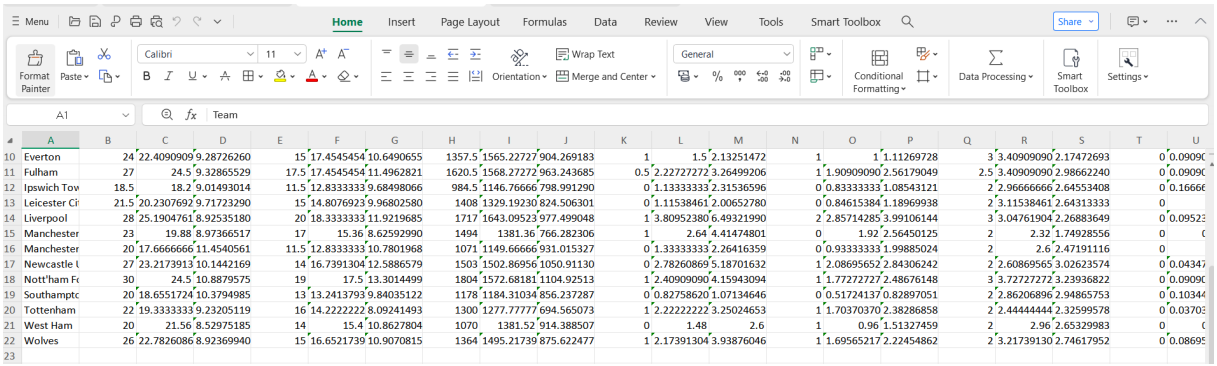
Step 6: Saving Results:

- Export the DataFrame final_results to the file results2.csv using the .to_csv() method.
- The parameter index=True is used (default) to save the DataFrame index (which is 'all' or the team name) to the first column of the CSV file.

2.2.4 Results

The program ran successfully and generated the file results2.csv. The results2.csv file contains the aggregated statistical results in the required format:

- Rows: The first row has the index 'all', representing statistics across all players. Subsequent rows have the index as the name of each team, representing statistics calculated specifically for the players of that team.
- Columns: The columns are named in the format "Metric of Statistic", for example: "Median of Performance: goals", "Mean of Performance: goals", "Std of Performance: goals", "Median of Performance: assists", etc., covering all metrics in stats_columns.
- Values: The cells contain the corresponding calculated median, mean, or standard deviation values. NaN values (if any, for example, the standard deviation of a group with only 1 player) will be represented as empty cells in the CSV file.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
10	Everton	24	22.4090909	9.28726260	15	17.4545454	10.6490655	1357.5	1565.22727	904.269183	1	1.5	2.13251472	1	1	1.11269728	3	3.40909090	2.17472693	0	0.0909(
11	Fulham	27	24.5	9.32865529	17.5	17.4545454	11.4962821	1620.5	1568.27272	963.243685	0.5	2.22727272	3.26499206	1	1.90909090	2.56179049	2.5	3.40909090	2.98662240	0	0.0909(
12	Ipswich Tow	18.5	18.2	9.01493014	11.5	12.8333333	9.68498066	984.5	1146.76666	798.991290	0	1.13333333	2.31536596	0	0.83333333	1.08543121	2	2.96666666	2.64553408	0	0.1666(
13	Leicester Ci	21.5	20.2307692	9.71723290	15	14.8076923	9.96802580	1408	1329.19230	824.506301	0	1.11538461	2.00652780	0	0.84615384	1.18969938	2	3.11538461	2.64313333	0	
14	Liverpool	28	25.1904761	8.92535180	20	18.3333333	11.9219685	1717	1643.09523	977.499048	1	3.80952380	6.49321990	2	2.85714285	3.99106144	3	3.04761904	2.26883649	0	0.0952:
15	Manchester	23	19.88	8.97366517	17	15.36	8.62592990	1494	1381.36	766.282306	1	2.64	4.41474801	0	1.92	2.56450125	2	2.32	1.74928556	0	(
16	Manchester	20	17.6666666	11.4540561	11.5	12.8333333	10.7801968	1071	1149.66666	931.015327	0	1.33333333	2.26416359	0	0.93333333	1.99885024	2	2.6	2.47191116	0	
17	Newcastle U	27	23.2173913	10.1442169	14	16.7391304	12.5886579	1503	1502.86956	1050.91130	0	2.78260869	5.18701632	1	2.08695652	2.84306242	2	2.60869565	3.02623574	0	0.0434:
18	Nott'ham Fo	30	24.5	10.8879575	19	17.5	13.3014499	1804	1572.68181	1104.92513	1	2.40909090	4.15943094	1	1.77272727	2.48676148	3	3.72727272	3.23936822	0	0.0909(
19	Southampto	20	18.6551724	10.3794985	13	13.2413793	9.84035122	1178	1184.31034	856.237287	0	0.82758620	1.07134646	0	0.51724137	0.82897051	2	2.86206896	2.94865753	0	0.1034(
20	Tottenham	22	19.3333333	9.23205119	16	14.2222222	8.09241493	1300	1277.77777	694.565073	1	2.22222222	3.25024653	1	1.70370370	2.38286858	2	2.44444444	2.32599578	0	0.0370:
21	West Ham	20	21.56	8.52975185	14	15.4	10.8627804	1070	1381.52	914.388507	0	1.48	2.6	1	0.96	1.51327459	2	2.96	2.65329983	0	(
22	Wolves	26	22.7826086	8.92369940	15	16.6521739	10.9070815	1364	1495.21739	875.622477	1	2.17391304	3.93876046	1	1.69565217	2.22454862	2	3.21739130	2.74617952	0	0.0869:
23																					

Figure 2.2: Partial view of results2.csv file

2.3 Visualizing Player Statistical Distribution with Histograms

2.3.1 Objective

The primary objective of the script plot_stat_histograms.py is to visualize and analyze the distribution of a set of key football player statistical indicators. Specifically, the script aims to:

- Generate histograms for each selected statistical indicator (stats_to_plot) to show its distribution across the entire league (all players).
- Create separate histograms for each football team, illustrating the distributio of each statistical indicator within that team.
- Save these histograms as image files (PNG) for easy review, sharing, and later analysis.

- Provide a visual insight into the shape of the distribution (e.g., symmetrical left-skewed, right-skewed), central tendency, and dispersion of the statistical indicators, both at the league level and the team level.

The specific statistical indicators analyzed in this script include: 'Performan goals', 'Performance: assists', 'Standard: shoots on target percentage (SoT%)', 'Blocks: Int', 'Performance: Recov', 'Challenges: Att'.

2.3.2 Library Selection

The script utilizes several popular Python libraries for data processing and visualization:

- pandas: The core library for reading data from CSV files (results1.csv), storing data as DataFrames, and performing necessary data processing, filtering, and selection for plotting. It also handles missing values (NaN).
- matplotlib.pyplot: The foundational library for creating plots in Python. It is used to create figures, axes, set titles, labels, and manage the layout of plots, especially when drawing multiple subplots for teams.
- seaborn: A library built on top of matplotlib, providing a higher-level interface for drawing attractive and informative statistical graphics. Specifically, seaborn.histplot is used to draw histograms, including a kernel density estimate (KDE) curve to smooth and better visualize the distribution shape. seaborn.set_theme is used to set the overall aesthetic style for the plots.
- numpy: The scientific computing library, used in the calculate_fd_bins function to perform necessary mathematical operations (such as cube root np.cbrt, base-2 logarithm np.log2) for calculating the optimal number of bins according to the Freedman-Diaconis and Sturges rules.
- os: The library providing functions for interacting with the operating system, mainly used for managing file paths (getting the current directory path, joining paths), creating storage directories (stat_histograms), and ensuring valid paths.
- math: The basic mathematics library, used for functions like math.ceil to round up when calculating the number of rows/columns for subplots and the number of bins.

2.3.3 Implementation Steps

Defining Helper Functions:

- ensure_dir: Ensures that the output directory exists.
- sanitize_filename: Cleans the statistical indicator name to create a valid file name.

- calculate_fd_bins: Calculates the optimal number of bins for the histogram based on the Freedman-Diaconis rule (preferred) or Sturges' rule (fallback), helping the plot better reflect the actual data structure and avoiding arbitrary bin selection. Limits the maximum number of bins to prevent over-detail.

Defining Plotting Functions:

- plot_overall_hist:
  - Receives a DataFrame, the statistical indicator name, and the output directory.
  - Filters data for the statistical indicator, converts it to numeric, and removes missing values (NaN).
  - Calculates the optimal number of bins using calculate_fd_bins.
  - Uses seaborn.histplot to draw the overall distribution histogram, enabling KDE.
  - Sets the title, axis labels, and saves the plot to a PNG file with a standardized name.
- plot_team_hist:
  - Receives a DataFrame, the indicator name, the output directory, the team column name, and the number of teams per plot.
  - Gets a list of unique teams and divides them into smaller groups (accordin to TEAMS_PER_PLOT).
  - Iterates through each group of teams:
    * Creates a new figure with subplots (a grid of smaller plots).
    * Filters the data to include only teams in the current group.
    * Calculates the optimal number of bins based on the data range of the teams in this group. This ensures that histograms within the same figure have the same binning.
    * Determines common x-axis limits (x_min, x_max) for all subplots in the current figure for easier comparison.
    * Draws a histogram for each team on a separate subplot using seaborn.his using the calculated number of bins and bin range (binrange).
    * Sets the subplot title to the team name, hides axis labels to avoid clutter, and applies consistent x-axis limits.
    * Hides any unused subplots.
    * Sets the main title for the figure (including the indicator name and team group).
    * Saves the figure containing the team group's histograms to a PNG file.

Main Execution Function (if __name__ == '__main__':)

- Sets the theme for seaborn.

- Defines the list of statistical indicators to plot (stats_to_plot).
- Defines the path and reads the results.csv file into a DataFrame.
- Ensures that the output directory exists.
- Iterates through each indicator in stats_to_plot:
  - Calls plot_overall_hist to plot the overall histogram.
  - Calls plot_team_hist to plot histograms for each team (grouped).

2.3.4 Limitations of Using Histograms

While histograms are a powerful visualization tool in football player statistical analysis, allowing us to quickly grasp the distribution of indicators (e.g., goals, playing time, assists), identify outliers, and compare players or teams, it's important to note that this method also has certain limitations. To avoid misleading or biased conclusions, we need to consider the following factors:

- Influence of Data Quality: The reliability of histograms directly depends on the completeness and accuracy of the input data (in the results.csv file). Removing missing values (NaN) during processing is necessary to ensure the accuracy of the plots. However, if there is a significant amount of missing data or if the missing data is not random, the shape of the distribution may be distorted, leading to subjective and inaccurate assessments of general trends.
- Sensitivity to the Number of Bins: Although the Freedman-Diaconis rule is applied to determine the optimal number of bins, aiming to represent data details in the best way, the shape of the histogram can still change significantly with variations in the number of bins. Additionally, imposing a maximum limit on the number of bins (via the MAX_FD_BINS variable) is subjective and has the potential to reduce the resolution of the histogram, especially when the data distribution has a large spread.

2.4 Performance Analysis of Premier League Teams 2024–2025

2.4.1 Introduction

To gain an in-depth understanding of the performance of teams in the 2024-2025 Premier League season, this analysis is structured around two main objectives. Firstly, to identify which team leads in each statistical category. Secondly, to provide an assessment of which team has the most impressive overall performance. To achieve this, we rely on information processed by two important software components:

- highest_stats_team.py: This is the heart of the analysis process - a Python script designed to automate everything. It handles tasks from reading input data, sorting indicators into different categories (e.g.,

positive, negative, or irrelevant indicators), and then determining which team has the best (or worst, as the case may be) performance in each category. Finally, it synthesizes all this information to draw a conclusion about the best-performing team of the season.

- config.py: This file acts as a control panel, allowing users to customize various aspects of the analysis process. For example, users can change the path to the data, the rules for handling missing data, or how to categoriz indicators. This flexibility makes the software easily reusable, reduces maintenance effort, and allows for the analysis of data from different seasons or different sources.

This approach not only saves time and effort but also provides flexibility and scalability for future analyses.

2.4.2 Main Ideas

- Identifying leading teams by each indicator: The program is designed to divide statistical indicators into three main groups (beneficial indicators adverse indicators, and ignored indicators) based on rules specified in the config.py file. For each indicator, the data is converted to numerical form, and then the program searches for the highest value (for beneficial indicators) or the lowest value (for adverse indicators). The team that achieves this value is identified as the leader in that category. To illustrate, the program can identify which team scored the most goals, which team had the highest successful pass rate, or which team received the fewest red cards.

- Ranking teams based on overall performance: The program will count the number of indicators in which each team leads, separately for beneficial and adverse indicators. It will then calculate a "performance score" for each team by adding the number of leads in both types of indicators. The team with the highest performance score will be considered the best-perfor team of the season.

2.4.3 Execution Process

The analysis process is carried out in a sequence of steps controlled by the highest_stats_team.py script:

1. Setting up and loading configuration parameters:

The configuration parameters are read from the config.py file through a function called load_config. This function is responsible for reading various configuration variables and creating sets from the IGNORE_STATS, BAD_STATS, and GOOD_STATS variables. These sets are used to classify statistical indicators in the subsequent steps.

2. Constructing the path and reading data from the CSV file:

The program creates the full path to the results2.csv file. Data from the CSV file is read into a DataFrame (a type of data table) using the pandas library. The rules for handling missing values are taken from the NA_VALUES variable in the config.py file.

3. Initial data processing in the DataFrame:

The column containing information about team names is identified (the first column is considered the default). The names of all teams are converted to lowercase, and data rows with the team name "all" are removed from the DataFrame.

4. Finding leading teams for each valid statistical indicator (Function main):

Step 1: Selecting statistical indicators for analysis:

The program creates a list including columns with the prefix "Mean of " (excluding the column containing team names). For each column in this list: The actual name of the indicator (removing "Mean of ") is extracted. The indicator is assigned a "type" (using the classify_stat function): 'ignore', 'bad', 'good', or 'uncategorized'. This type assignment is based on the keyword sets loaded in the previous step. If the indicator is not ignored, the data in that column is converted to numerical form. Only columns with at least one valid numerical value are retained for further analysis.

Step 2: Detailed analysis of each selected indicator:

For each indicator retained in the previous step: The data in the indicator column is converted to numerical form, and any invalid values (NaN) are removed. The program determines the "best" value: If the indicator is of type 'good' or 'uncategorized': the program searches for the largest value. If the indicator is of type 'bad': the program searches for the smallest value. If the program does not find the best value (e.g., the column contains only NaN values), it records an error message. Otherwise (if the best value is found): The program identifies the team(s) that achieved this best value. Information about the leading team(s), the best value (kept in its original format for display), and the type of indicator are stored in a data structure called a dictionary.

Step 3: Returning the aggregated results:

The function returns a dictionary containing all the analysis results for each indicator.

5. Presenting detailed analysis results:

The results are sorted by the name of the statistical indicator. For each analyzed indicator, the program formats the value (using the format_value function to handle integers, floats, and the string "N/A") and prints the following information: the name of the indicator, the type of indicator (GOOD, BAD), the leading team(s), and the best value achieved by that team.

6. Calculating and displaying the overall ranking:

Step 1: Counting the number of leads:

For each indicator with valid results in the analysis_results data structure: If the leading team's indicator type is 'good', the lead count for that team in the good_lead_counts data structure is incremented. If the leading team's indicator type is 'bad', the lead count for that team in the bad_lead_counts data structure is incremented.

Step 2: Calculating the overall performance score:

The program collects a list of all distinct teams present in the DataFrame. The initial score for each team is set to 0. The overall performance score for each team is calculated by adding the total number of leads in 'good' and 'bad' indicators.

Step 3: Sorting and displaying the ranking:

The teams are sorted in descending order of their overall performance score.

If two teams have the same score, they are sorted alphabetically by team name. The final ranking is printed, including each team's rank, team name, and total number of leads.

2.4.4 Results

When the highest_stats_team.py program finishes running, it will produce two main pieces of information, displayed directly on the screen. These results provide a comprehensive view of each team's performance in each analyzed statistica indicator, as well as an overall ranking of the teams based on the number of times they achieved the leading position in those indicators.

Example of detailed analysis results for each indicator:

```
Teams with the highest (or lowest for bad stats) value for each statistic:
Performance: goals [GOOD]: Liverpool (value: 28)
Performance: assists [GOOD]: Liverpool (value: 18)
Performance: yellow cards [BAD]: Leicester City (value: 0)
Performance: red cards [BAD]: West Ham (value: 0)
Expected: expected goals (xG) [GOOD]: Liverpool (value: 24.0)
Expected: expedted Assist Goals (xAG) [GOOD]: Liverpool (value: 13.2)
Progression: PrgC [GOOD]: Manchester City (value: 189)
Progression: PrgP [GOOD]: Manchester Utd (value: 281)
Progression: PrgR [GOOD]: Liverpool (value: 440)
Per 90 minutes: Gls [GOOD]: Aston Villa (value: 0.99)
```

Figure 2.3: Example image of detailed analysis results for each indicator

Example of the overall team ranking:

```
Summary ranking (number of times leading in statistics):
Liverpool: 17 top statistics
West Ham: 5 top statistics
Southampton: 5 top statistics
Manchester City: 4 top statistics
Manchester Utd: 4 top statistics
Chelsea: 4 top statistics
Brighton: 4 top statistics
Brentford: 4 top statistics
Bournemouth: 2 top statistics
```

Figure 2.4: Example image of the overall team ranking

Statistical data reveals Liverpool as a team with incredible strength and comprehensiveness. They not only excel in a few specific aspects but also maintain stable performance across most important modern football metrics. This team possesses formidable attacking capabilities, superior ball control,

along with a solid defensive system - creating a cohesive and effective playing unit. In fact, Liverpool definitively secured the Premier League title for the 2024-2025 season with 4 matches remaining, finishing with 82 points and leaving the runner-up Arsenal behind by a comfortable margin. This is clear evidence that when a team operates well both technically and strategically, their success will be clearly reflected in the statistics and on-field results.

Chapter 3

Player Clustering using K-Means and PCA

3.1 Introduction and Data Preparation

3.1.1 Introduction

This report details the application of the K-means clustering algorithm to classify football players into groups based on their performance statistics. The main objectives of this process are:

- To determine the optimal number of clusters (i.e., the best number of groups) into which the players should be divided.
- To perform player clustering based on the determined number of clusters.
- To visualize these clusters on a two-dimensional plot using the Principal Component Analysis (PCA) dimensionality reduction technique.

3.1.2 Data Preparation

To prepare for the analysis, player statistics data was processed through the following steps:

- Load Data: The data was loaded from a CSV (Comma Separated Values) file, which is a common format for storing tabular data.
- Select Relevant Data Columns: Columns containing statistical information about player performance (e.g., number of goals, number of assists, number of tackles, etc.) were selected for analysis. Columns not related to statistics (e.g., player name, nationality, team name) and the team information column were removed.
- Handle Missing Values: During data collection, there may be cases where some information is missing. To ensure the integrity of the analysis, these missing values were filled in with the mean value of the column containing them.

- Data Scaling: Different statistical indicators may have different scales and units. To ensure that every indicator contributes equally to the clustering process, the data was "standardized" using StandardScaler. This process transforms the data so that it has a mean of 0 and a standard deviation of 1.

3.2 Clustering and Dimensionality Reduction

3.2.1 Determining the Optimal Number of Clusters k

To determine the optimal number of clusters (k), the Elbow method was used. This method works as follows:

- Inertia: For each value of k (number of clusters), the K-means algorithm is run to partition the data into clusters. "Inertia" is the sum of the squared distances of each data point to its nearest cluster center. The smaller the inertia, the closer the points within a cluster are to each other, meaning the clusters are more compact.
- Elbow Plot: A graph is drawn, with the x-axis representing the number of clusters (k) and the y-axis representing the corresponding inertia value.
- Finding the "Elbow" Point: The shape of the graph often resembles an arm, and the "elbow" point is the point where the slope of the curve changes abruptly. This point indicates that increasing the number of clusters beyond that point does not significantly reduce the inertia, meaning it does not substantially improve the compactness of the clusters.

In this study:

- The tested values of k ranged from 1 to 10. The maximum value of 10 was chosen to limit computational complexity while still allowing for the exploration of a reasonable number of clusters.
- The Elbow plot (elbow_plot.png) clearly shows the largest change in slope at k = 6.
- Therefore, the optimal number of clusters chosen is 6.

K-Means Clustering

- The K-means algorithm was run with the number of clusters (k) set to 6.
- This algorithm divides the players into 6 distinct clusters, such that each player is assigned to the cluster whose centroid is closest to them.
- These clusters are labeled from 0 to 5.

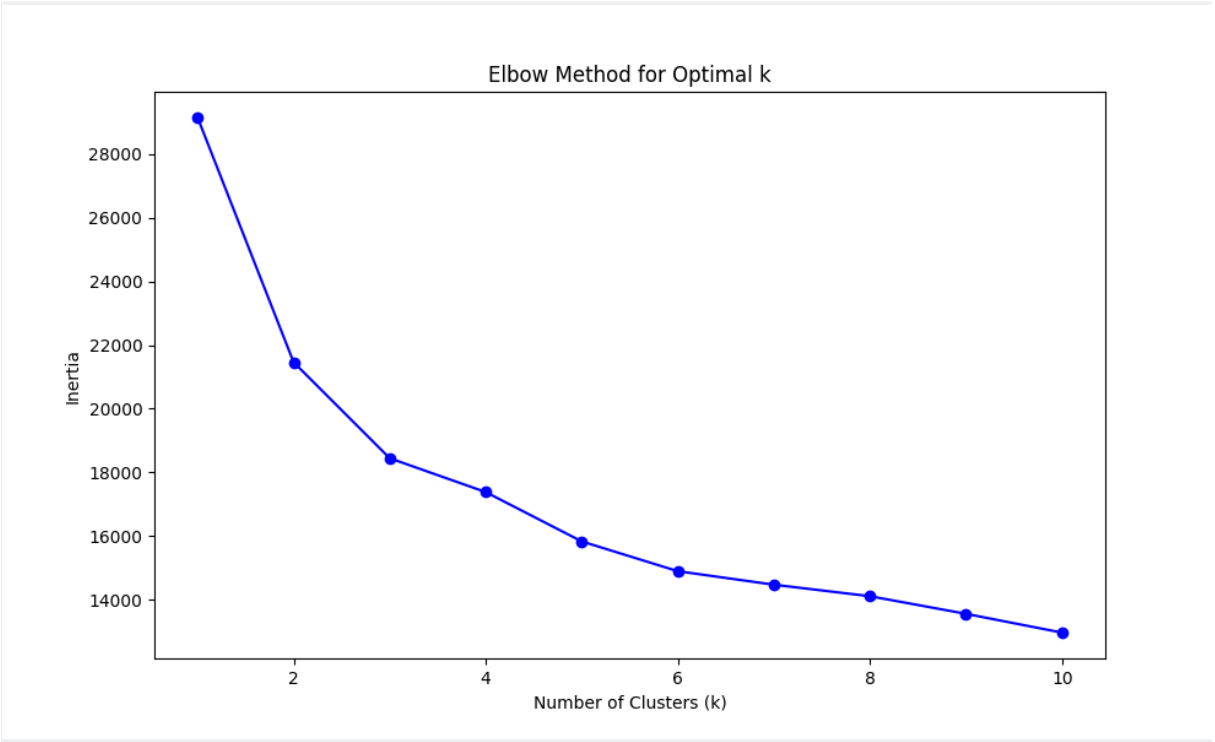


Figure 3.1: Elbow Plot Image

3.2.2 Dimensionality Reduction with PCA

- The initial player data had many dimensions (each dimension corresponding to a statistical metric). To enable visual representation on a 2D plot, it is necessary to reduce the dimensionality of the data.
- Principal Component Analysis (PCA) technique was used to reduce the number of dimensions to 2. PCA finds two "principal components" (PC1 and PC2) which are two directions that combine the original metrics in a way that retains the most information from the data.
- The "explained variance ratio" indicates the percentage of the original data's information (variance) that is retained in these two principal components. The higher this value, the more accurately the 2D plot represents the original data.

3.2.3 Visualization of Clusters

- A 2D scatter plot (player_clusters.png) was created, with PC1 as the x-axis and PC2 as the y-axis.
- Each point on the plot represents a player.
- The color of each point indicates the cluster to which that player is assigned.
- The colors correspond to the cluster labels as follows:
  - Blue: Cluster 0

- Orange: Cluster 1
- Green: Cluster 2
- Red: Cluster 3
- Purple: Cluster 4
- Brown: Cluster 5

3.2.4 Cluster Analysis

To better understand the meaning of each cluster, a detailed analysis was performed

- Cluster 0 (Blue): Good Passing Wing-backs
  - *Position:* This cluster mainly includes full-backs and wing-backs.
  - *Key Characteristics:*
    - Players in this cluster tend to make a large number of passes, especially short and medium passes, with high accuracy.
    - They also frequently participate in ball circulation and moving the ball up the field.
    - Their defensive stats are average, indicating they contribute to both attack and defense.
- Cluster 1 (Orange): Well-Rounded Central Midfielders
  - *Position:* This cluster focuses on central midfielders.
  - *Key Characteristics:*
    - Players in this cluster have a balance between attacking and defensive capabilities.
    - They participate in ball recovery, interceptions, and initiating attacks, passing in the midfield area.
    - They have good passing ability, both short and medium range, and also make a certain contribution to creating scoring opportunities.
- Cluster 2 (Green): Technical Wing Attackers
  - *Position:* This cluster includes wing midfielders and wing forwards.
  - *Key Characteristics:*
    - Players in this cluster stand out with their dribbling skills, ability to create breakthroughs, and make crosses into the opponent's penalty area.
    - They tend to score goals and create chances for teammates.
    - Defensive stats are not their strong point.
- Cluster 3 (Red): Strong Defensive Midfielders
  - *Position:* This cluster focuses on defensive midfielders and center-backs.
  - *Key Characteristics:*

        * Players in this cluster have strong defensive capabilities, with high stats in tackles, interceptions, and duels.
        * They mainly focus on recovering the ball and protecting the defense.
        * Their attacking and assisting abilities are usually more limited.

- Cluster 4 (Purple): Target Forwards
    - *Position:* This cluster mainly includes center forwards.
    - *Key Characteristics:*
        * Players in this cluster tend to play near the opponent's goal and participate in aerial duels.
        * They have good scoring ability, especially from shots within the penalty area.
        * They may not participate much in building up play.

- Cluster 5 (Brown): Goalkeepers
    - *Position:* This cluster includes goalkeepers.
    - *Key Characteristics:*
        * Their most important stats are related to saving, catching, and reflexes.
        * Their attacking and passing stats are very low.

The CSV files cluster_0_players.csv, cluster_1_players.csv, cluster_2_players.csv, cluster_3_players.csv, cluster_4_players.csv and cluster_5_players.csv contain detailed data about the corresponding clusters.

3.3 Conclusion

The K-means algorithm has been successfully applied to classify players into 6 groups based on statistical indicators. The Elbow method provides a data-driven approach to selecting the optimal number of clusters. PCA helps simplify the visualization of clusters in a 2D space. Detailed cluster analysis sheds light on the key statistical characteristics that distinguish the different player groups.

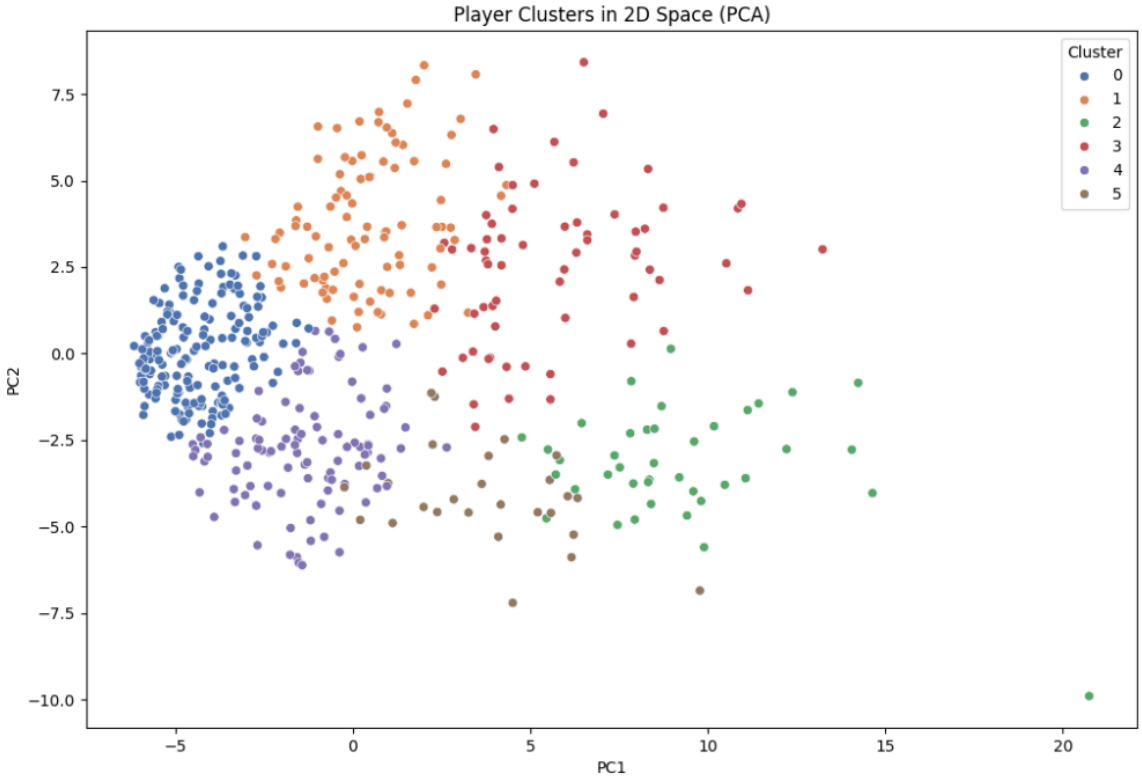


Figure 3.2: 2D Scatter Plot Image

Chapter 4

Player Value Estimation

4.1 Introduction and Data Collection Method

4.1.1 Introduction

The football player transfer market is a dynamic economic sector where player values play a crucial role in transactions. Accurately estimating this value offers significant benefits to clubs in asset valuation, contract negotiations, as well as to agents and other stakeholders. This chapter focuses on building and evaluating a method for estimating player transfer values based on data collected from the website https://www.footballtransfers.com. The main objective is to provide a quantitative support tool for making more informed decisions in the transfer market, rather than relying solely on intuition or personal experience. This method combines statistical elements of player performance, personal information, and club factors to create a comprehensive prediction model.

4.1.2 Data Collection

Player transfer value data for the 2024-2025 season was collected from the website https://www.footballtransfers.com using web scraping techniques. The data collection process was automated to gather information from multiple pages and players. To ensure the representativeness and reliability of the data, only players with more than 900 minutes of playing time were included in the analysis. This helps to eliminate players with minimal actual contribution and focuses on those who play regularly.

4.2 Player Value Estimation Method

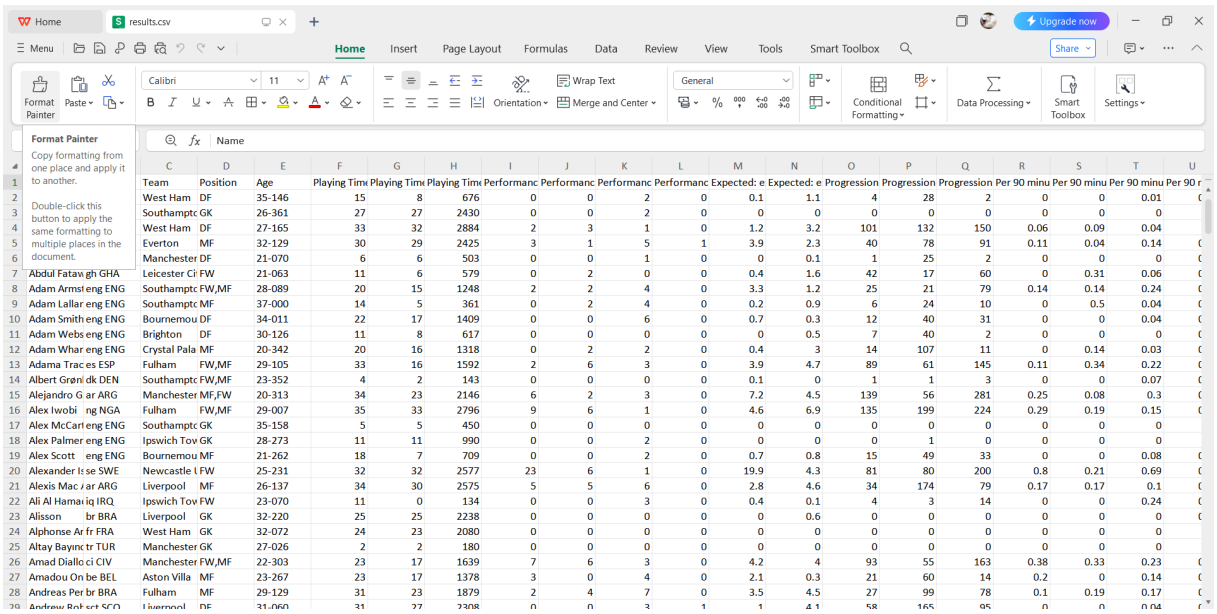
4.2.1 Feature Selection

The features selected for estimating player value include:

- Personal information: Age, playing position
- Performance statistics: Number of minutes played, goals, assists, etc. (from the results.csv file)

– The results.csv file contains detailed statistical data on player performance during the season, collected from the previous problem (Problem 1). These indicators reflect the player's ability and contributi on the field.

- Club information: Current club



	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Team	Position	Age	Playing Time	Playing Time	Playing Time	Performanc	Performanc	Performanc	Performanc	Expected: e	Expected: e	Progression	Progression	Progression	Per 90 minu	Per 90 minu	Per 90 minu	Per 90 r
2	West Ham	DF	35-146	15	8	676	0	0	2	0	0.1	1.1	4	28	2	0	0	0.01	
3	Southamptc	GK	26-361	27	27	2430	0	0	2	0	0	0	0	0	0	0	0	0	
4	West Ham	DF	27-165	33	32	2884	2	3	1	0	1.2	3.2	101	132	150	0.06	0.09	0.04	
5	Everton	MF	32-129	30	29	2425	3	1	5	1	3.9	2.3	40	78	91	0.11	0.04	0.14	
6	Manchester	DF	21-070	6	6	503	0	0	1	0	0	0.1	1	25	2	0	0	0	
7	Abdul Fatawgh GHA	Leicester Ci	FW	21-063	11	6	579	0	2	0	0	0.4	1.6	42	17	60	0	0.31	0.06
8	Adam Armsteng ENG	Southamptc	FW,MF	28-089	20	15	1248	2	2	4	0	3.3	1.2	25	21	79	0.14	0.14	0.24
9	Adam Lallar eng ENG	Southamptc	MF	37-000	14	5	361	0	2	4	0	0.2	0.9	6	24	10	0	0.5	0.04
10	Adam Smith eng ENG	Bournemou	DF	34-011	22	17	1409	0	0	6	0	0.7	0.3	12	40	31	0	0	0.04
11	Adam Webs eng ENG	Brighton	DF	30-126	11	8	617	0	0	0	0	0	0.5	7	40	2	0	0	0
12	Adam Whar eng ENG	Crystal Pala	MF	20-342	20	16	1318	0	2	2	0	0.4	3	14	107	11	0	0.14	0.03
13	Adama Trac es ESP	Fulham	FW,MF	29-105	33	16	1592	2	6	3	0	3.9	4.7	89	61	145	0.11	0.34	0.22
14	Albert Grøn dk DEN	Southamptc	FW,MF	23-352	4	2	143	0	0	0	0	0.1	0	1	1	3	0	0	0.07
15	Alejandro G ar ARG	Manchester	MF,FW	20-313	34	23	2146	6	2	3	0	7.2	4.5	139	56	281	0.25	0.08	0.3
16	Alex Iwobi ng NGA	Fulham	FW,MF	29-007	35	33	2796	9	6	1	0	4.6	6.9	135	199	224	0.29	0.19	0.15
17	Alex McCarl eng ENG	Southamptc	GK	35-158	5	5	450	0	0	0	0	0	0	0	0	0	0	0	0
18	Alex Palmer eng ENG	Ipswich Tov	GK	28-273	11	11	990	0	0	2	0	0	0	0	1	0	0	0	0
19	Alex Scott eng ENG	Bournemou	MF	21-262	18	7	709	0	0	2	0	0.7	0.8	15	49	33	0	0	0.08
20	Alexander Is se SWE	Newcastle U	FW	25-231	32	32	2577	23	6	1	0	19.9	4.3	81	80	200	0.8	0.21	0.69
21	Alexis Mac Ar ARG	Liverpool	MF	26-137	34	30	2575	5	5	6	0	2.8	4.6	34	174	79	0.17	0.17	0.1
22	Ali Al Hamaiq IRQ	Ipswich Tov	FW	23-070	11	0	134	0	0	3	0	0.4	0.1	4	3	14	0	0	0.24
23	Alisson br BRA	Liverpool	GK	32-220	25	25	2238	0	0	0	0	0	0.6	0	0	0	0	0	0
24	Alphonse Ar fr FRA	West Ham	GK	32-072	24	23	2080	0	0	0	0	0	0	0	0	0	0	0	0
25	Altay Bayınc tr TUR	Manchester	GK	27-026	2	2	180	0	0	0	0	0	0	0	0	0	0	0	0
26	Amad Diallo ci CIV	Manchester	FW,MF	22-303	23	17	1639	7	6	3	0	4.2	4	93	55	163	0.38	0.33	0.23
27	Amadou On be BEL	Aston Villa	MF	23-267	23	17	1378	3	0	4	0	2.1	0.3	21	60	14	0.2	0	0.14
28	Andreas Per br BRA	Fulham	MF	29-129	31	23	1879	2	4	7	0	3.5	4.5	27	99	78	0.1	0.19	0.17
29	Andrew Rob sct SCO	Liverpool	DF	31-060	31	27	2308	0	0	3	1	1	4.1	58	165	95	0	0	0.04

Figure 4.1: Image of results.csv file

The selection of these features is based on the assumption that they significantly influence a player's value in the transfer market. For example, age and playing position often affect a player's potential and remaining playing time, while playing statistics reflect their actual performance and efficiency.

4.2.2 Model Selection

The Gradient Boosting Regression (GBR) model is used to estimate player value. GBR is a powerful algorithm capable of capturing complex non-linear relationships between features and the target variable (transfer value). Compared to simpler methods like linear regression, GBR can model intricate interactions between variables and provide more accurate predictions.

4.2.3 Analysis of Visualization Images of Model Performance

To visualize and better understand the model's performance, the following three images are used:

- $\texttt{feature}_{i}mportance.png$: $Achartshowingtheimportancelevelofinputfeaturesinpredictingtransfer$

Chart of Input Feature Importance ($\texttt{feature}_{i}mportance.png$)

This chart shows the features that play the most important role in predicting transfer value. Features with a higher importance level have a greater impact on the model's prediction results.

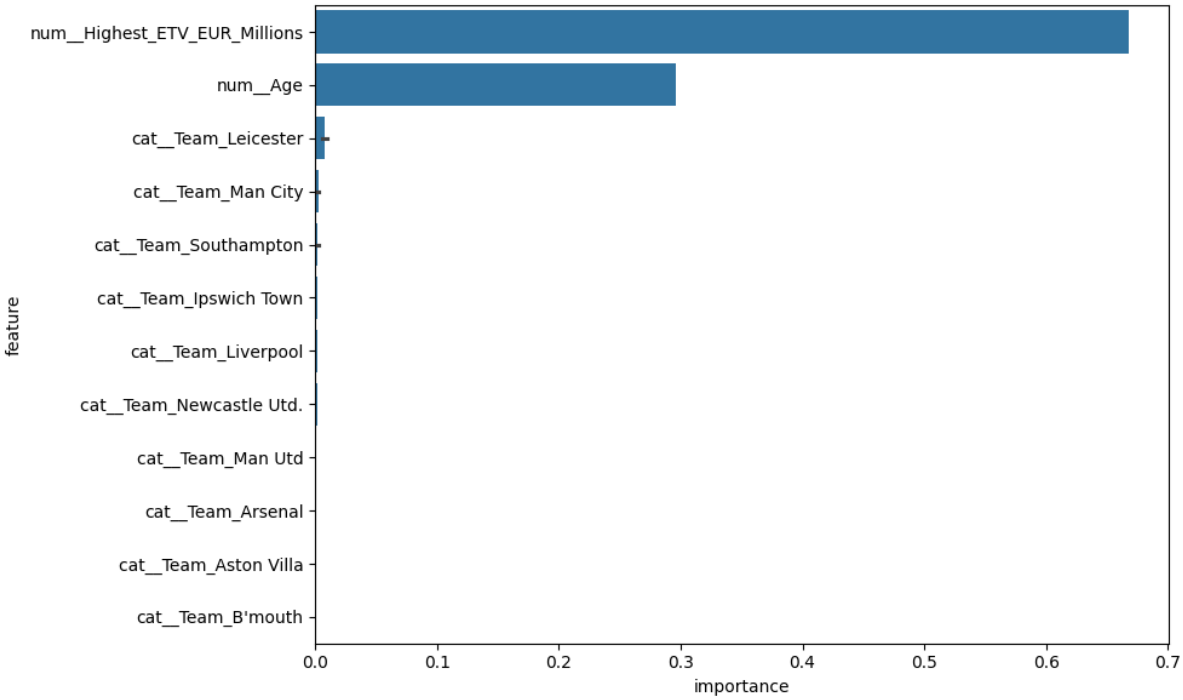


Figure 4.2: Image of the feature$_i mportancechart$

Observations:

- The specific names of the features need to be identified from the chart (e.g., "Age", "Position", "Goals", "Assists", etc.).
- Analyze the order and the difference in importance levels between the features.
- Explain the significance of each feature's importance in the football context (e.g., "Number of goals" may be important because it reflects a player's scoring ability and commercial appeal).

Chart Comparing Predicted and Actual Values (pred$_v s_a ctual.png$)

This chart displays the relationship between the transfer value predicted by the model and the actual transfer value. Data points close to the diagonal line (y = x) indicate accurate model predictions.

Observations:

- Evaluate the dispersion of data points around the diagonal line.
- Comment on the model's tendency (e.g., does it tend to over- or under-estimate the value of certain players?).
- Identify any outliers (if any) and speculate on the reasons (e.g., a player with an unusually high value due to commercial factors).

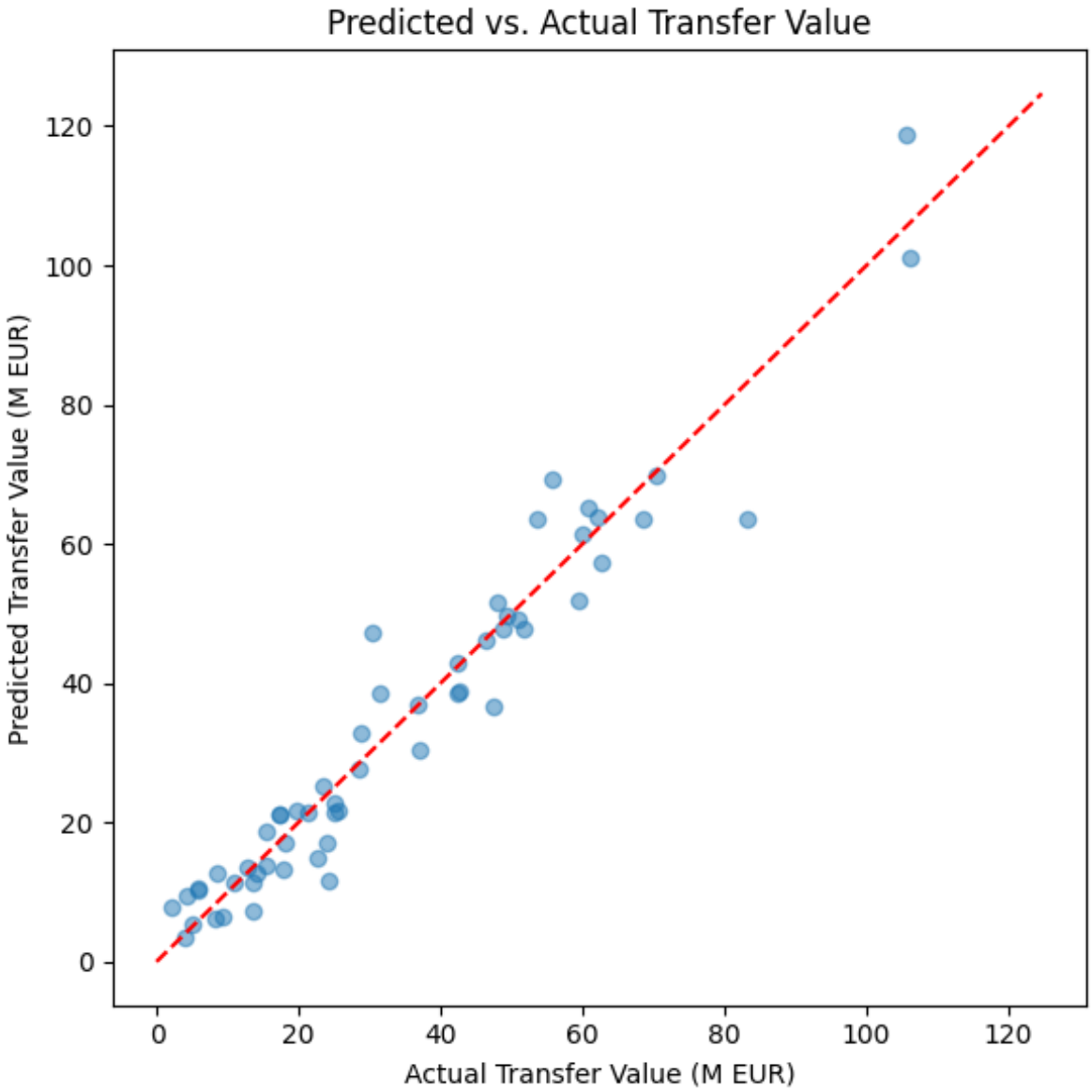


Figure 4.3: Image of the $\text{pred}_v s_a ctualchart$

Residual Plot (`residuals.png`)

This chart displays the distribution of residuals, which is the error between the actual and predicted values. A good model has residuals that are randomly distributed around 0, with no clear trends.

Observations:

- Evaluate the symmetry and distribution of the residuals (e.g., is it bell-shape
- Check if the residuals tend to increase or decrease with the predicted values.
- Identify any patterns in the residuals (if any) and speculate on the reasons (e.g., the model may be overlooking an important factor).

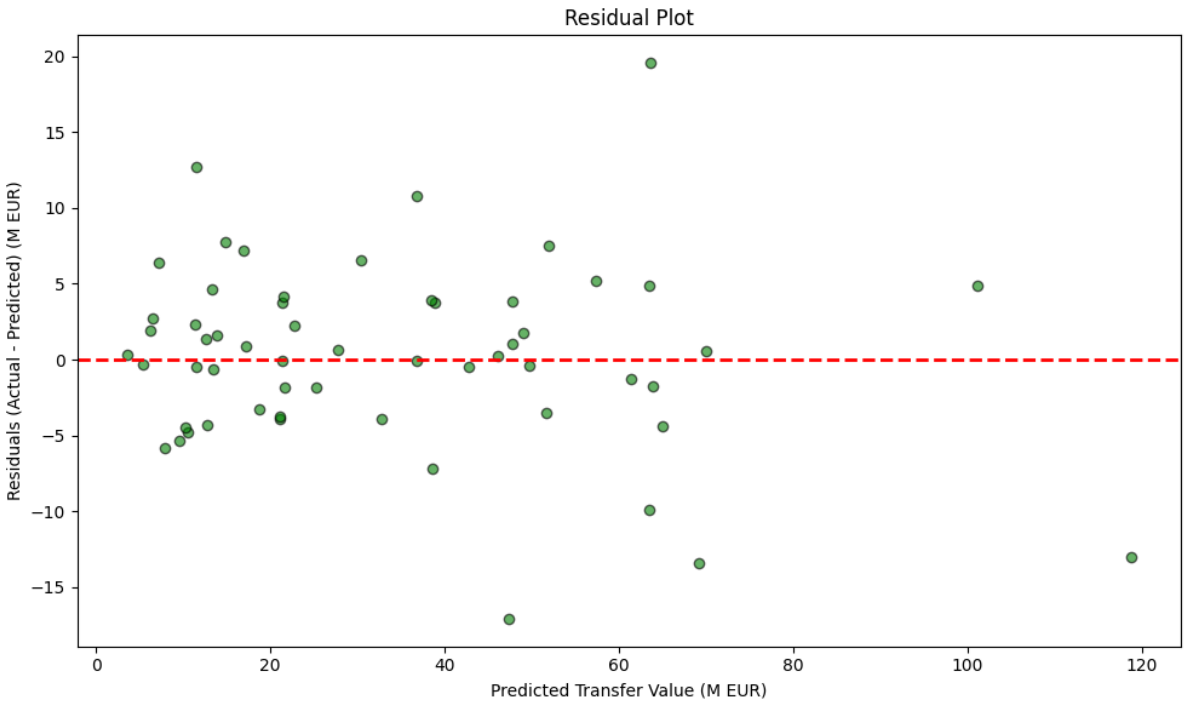


Figure 4.4: Image of the residuals chart

4.2.4 Model Training and Evaluation

- The data was split into a training set (80%) and a test set (20%) to evaluate the model's generalization ability.
- The GBR model was trained on the training set using 5-fold cross-validation and RandomizedSearchCV to optimize the parameters. Cross-validation helps evaluate the model on different subsets of the data, while RandomizedSearchCV searches for the best parameters for the model.
- The model's performance was evaluated on the test set using the R2, RMSE, and MAE metrics.

4.2.5 Results

```
[Running] python -u "d:\Python\BTL\SourceCode\Ex4\estimate.py"
Fitting 5 folds for each of 30 candidates, totalling 150 fits
R2: 0.937, RMSE: 6.11, MAE: 4.43
```

Figure 4.5: Model evaluation results

The model training and evaluation results show:

- Fitting 5 folds for each of 30 candidates, totalling 150 fits: The model optimization process tested 30 different sets of parameters across 5 data folds (5-fold cross-validation), resulting in a total of 150 model training and evaluation runs. The cross-validation technique helps ensure the

stability and reliability of the results, avoiding overfitting (the model only performs well on the training data).

- R2: 0.937: The coefficient of determination (R2) is a statistical measure that indicates how well the model fits the data. The R2 value ranges from 0 to 1.
  - R2 = 1: The model predicts perfectly; all variations in the target variable (transfer value) are explained by the input features.
  - R2 = 0: The model has no predictive power and does not explain any variation in the target variable.
  - R2 = 0.937: Our model explains 93.7% of the variation in transfer value, indicating a very good fit to the data. This means the model can predict player transfer values accurately based on the selected features.
- RMSE: 6.11: Root Mean Squared Error (RMSE) is a metric that measures the average magnitude of the errors between the actual and predicted values. The smaller the RMSE, the more accurate the model.
  - RMSE = 6.11: The average error between the actual and predicted values is 6.11 million EUR. This is a relatively low error compared to the actual range of transfer values, indicating high model accuracy.
- MAE: 4.43: Mean Absolute Error (MAE) is also an error measurement metric, but unlike RMSE, MAE calculates the average of the absolute errors (without squaring). MAE is less sensitive to outliers than RMSE.
  - MAE = 4.43: The average absolute error between the actual and predicted values is 4.43 million EUR. The low MAE value further confirms the model's accuracy.

4.2.6 Limitations

The player value estimation model achieves impressive performance; however, some limitations still exist:

- Lack of qualitative factors: The current model primarily relies on quantitati factors (statistics, age), neglecting important qualitative factors such as player reputation, international experience, leadership skills, etc.
- Generalization ability: The model was trained on data from a specific league (Premier League); therefore, its applicability to other leagues may be limited due to differences in the football environment and transfer market.
- Prediction error: Although the evaluation metrics are good, there is still a certain level of error in the predictions. This may be due to the complexity of the transfer market and the influence of random factors.

4.3 Conclusion and Recommendations

The GBR model is a useful tool for estimating player transfer values, outperforming simpler methods like linear regression due to its ability to handle non-linear relationships. However, to enhance the model's accuracy and practical application, the following recommendations should be considered:

- Incorporate qualitative factors: Research methods to quantify qualitative factors (e.g., using sentiment analysis to assess player reputation from news and social media).
- Expand the scope of application: Collect data from various leagues and build separate models or a multi-task model to increase generalization ability.
- Improve the model: Experiment with other machine learning algorithms (e.g., Random Forest, Neural Networks) and combine them to create a more robust ensemble model.

Bibliography

[1] McKinney, W. (2017). *Python for Data Analysis*. O'Reilly Media. Accessed May 6, 2025, from https://wesmckinney.com/pages/book.html

[2] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras TensorFlow*. O'Reilly Media.

[3] Selenium Developers. (2024). *Selenium Documentation*. Accessed May 6, 2025, from https://www.selenium.dev/documentation/

[4] BeautifulSoup Developers. (2024). *Beautiful Soup Documentation*. Accessed May 6, 2025, from https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[5] Scikit-learn Developers. (2024). *Scikit-learn User Guide*. Accessed May 6, 2025, from https://scikit-learn.org/stable/user_guide.html

[6] Healy, K. (2019). *Data Visualization with Python*. O'Reilly Media.